BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Mx LASER PROCESSING SYSTEMS

Applications Manual

Rev 9446

GENERAL SCANNING INC.
TLSI Division
32 Cobble Hill Road, Somerville Ma. 02143
Tel. (617) 625-5200 FAX (617) 628-7966

P/N 273.328.00

LIMITED REPRODUCTION RIGHTS

For GENERAL SCANNING, INC./TLSI Customers

This document may be reproduced by a GSI customer under the Software Product Support Agreement solely for use by the customer's employees whose responsibilities include GSI equipment. Any copy of this document, or portion thereof, must contain the copyright and proprietary rights notice as stated on the original.

Developed and printed in U.S.A.

The material in this document is subject to change without notice. GSI assumes no responsibility for any errors which may appear in this document.

This document contains trade secrets and confidential information, and is furnished pursuant to a license to the user from TLSI. Use or reproduction of this document is restricted under the terms of the license.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (b) (3) (ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

GENERAL SCANNING, INC.
TLSI DIVISION
32 Cobble Hill Road
Somerville, MA 02143

LINKSPRINT

LinkSprint is the name of TLSI's high-speed link-blasting implementation. It permits the memory repair or logic programming application in an Mx system to cut links (fuses) as the beam positioner moves at a constant velocity. In older software releases and TLSI M118 laser systems, the beam positioner galvanometers would move to each link and be allowed to settle before the laser was pulsed. The settling time on older systems would be on the order of 20 to 50ms and would severely limit the throughput of the machine as the number of links to be cut increased.

The LinkSprint software moves the galvos to a group of links in a single axis and then sweeps across all the links and cuts only the desired ones. In this fashion, maximum link- cutting speeds of 4500 links per second may be achieved. The link group geometry determines the galvo velocity as well as the laser Q-switch rate. The laser is triggered at a constant frequency as the galvos are swept at a constant velocity such that a laser pulse is emitted over every link position. However, the Acousto-Optic Modulator (AOM) suppresses the pulse unless the link is programmed to be cut. Rapid cutting rates are due to the fast response of the AOM and the galvos' ability to move at positionally-corrected speeds exceeding 100µm/ms.

LinkSprint can only cut links when the link groups are parallel to the X or Y axis. Diagonal Link-Sprint groups are not allowed. There is no upper limit to the number of links that a device may have or to the number of LinkSprint groups within the device. However, the number of links within a group cannot exceed 6500. This is due to the memory size of the digital signal processor (DSP) which controls the galvos. Most applications rarely exceed several hundred LinkSprint groups, with two or three hundred links per group.

The general-purpose link-blasting application PROCESS_TOOL (see Section 1) requires link position information to be expressed in LinkSprint syntax. The coordinates of all the links contained in the device must be known beforehand and a numbering system agreed upon. The numbering of the links starts at 1 and ends at N where N is the total number of links that can be cut.

For example, consider a very simple device with only five links (all coordinates are in microns):

Link 1, X = 1545.7 Y = 338.92
Link 2, X = 1558.2 Y = 338.92
Link 3, X = 1570.7 Y = 338.92
Link 4, X = 1583.2 Y = 338.92
Link 5, X = 1595.7 Y = 338.92

The links expressed in LinkSprint syntax would be as follows:

```
5  1
1545.7    338.92    12.5    0.0    5
{ 1 2 3 4 5 }
```

The first line contains the numbers 5 and 1. . Five is the total number of links in the device and 1 is the total number of LinkSprint groups. The following two lines are interpreted as follows:

```
Xs  Ys  dx  dy  n
{ a b c d e f }
```

Xs and Ys are the coordinates of the first link in this group (1545.7, 338.92), dx is the space between the links in the x direction (x-pitch), and dy is the y-pitch. In this case, the x-pitch is 12.5 and the y-pitch is 0.0. n is the total number of links in this group which is 5 in this example. On the next line, the curly braces ({ }) enclose the *link number list.*

In this simple case, the first link (a) is link #1, the second link (b) is link #2, the third link (c) is link #3, and so on. The cuts would proceed from left to right if any of the links in this group are destined to be removed. Another method of expressing exactly the same link group is as follows:

```
5  1
1595.7     338.92     -12.5     0.0     5
{ 5 4 3 2 1 }
```

In this case, the LinkSprint group starts at link #5 and the cuts proceed from right to left. Note how the link numbers have been reversed. This ensures that any link number refers to the same link position. These examples are illustrated in Figure 1.

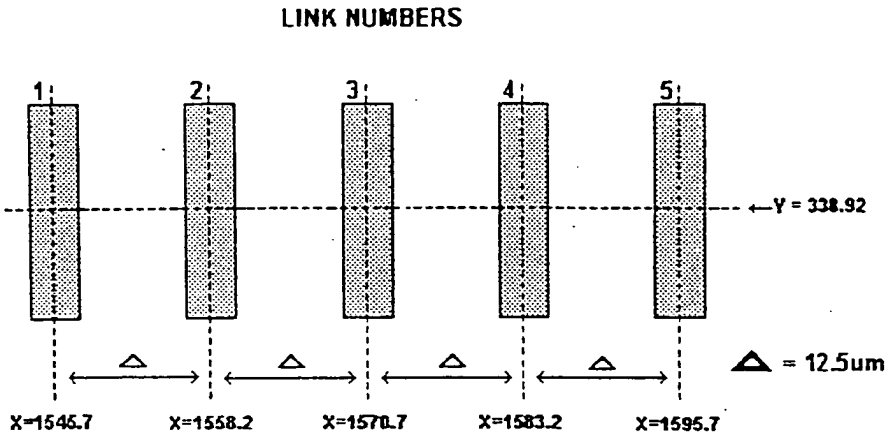

FIGURE 1.

Consider now a slightly more complex device with the following links:

```
Link 1,  X = -1547.0     Y = 8.5
Link 2,  X = -1569.0     Y = 8.5
Link 3,  X = -1558.0     Y = 8.5
Link 4,  X = -1580.0     Y = 8.5
Link 5,  X = -1591.0     Y = 8.5
Link 6,  X = 13.5        Y = 180.5
Link 7,  X = 13.5        Y = 200.5
Link 8,  X = 13.5        Y = 190.5
Link 9,  X = 13.5        Y = 210.5
```

A LinkSprint file could look like this:

```
9  2
-1591.0    8.5     11.0     0.0   5
{ 5 4 2 3 1}
13.5     180.5      0:0  10.0    4
{ 6 8 7 9 }
```

The first line indicates that a total of 9 links have been defined in two groups. Group 1 has a starting position of -1591.0,8.5; an x-pitch of 11.0; a y-pitch of 0.0 and a total of 5 links. The second line describes the link number assignment within the group. The first link in this group is link #5, the second link is link #4, the third link is link #2, and so on. Group 2 has a starting position of 13.5,180.5; an x-pitch of 0.0; a y-pitch of 10 and a total of 4 links. There are 4 links in group 2. The first link is link #6, the next one is link #8, and so on.

In order to further optimize the total throughput, the last link within a LinkSprint group should be as close as possible as the first link of the next group. This is the case in the previous example. These links are illustrated in Figure 2.

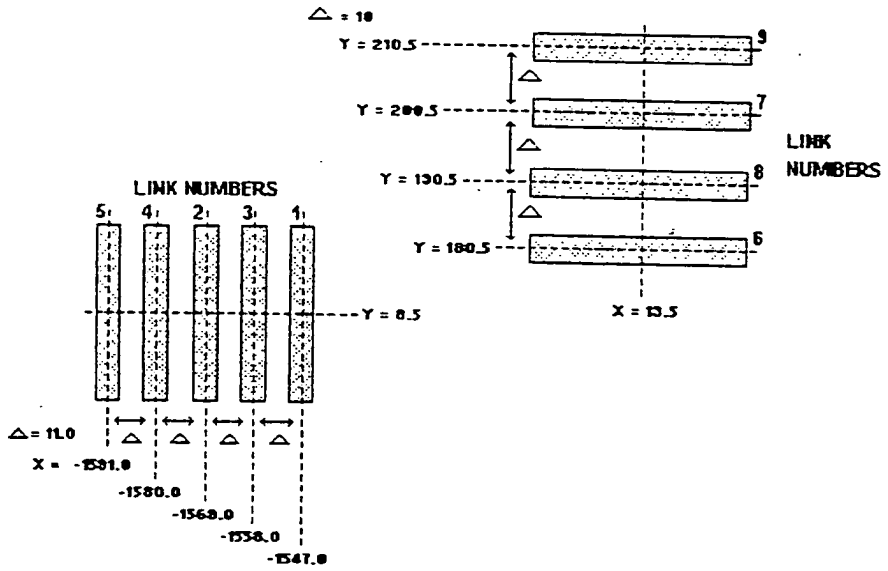

FIGURE 2.

There are three other important cases to consider: interleaved links, phantom links and single links. When patterns of two or more links are repeated at a constant pitch, the links are said to be interleaved. Consider for example the following links:

Link 17,	X = 0.0	Y = 1100.0
Link 21,	X = 3.5	Y = 1100.0
Link 313,	X = 7.0	Y = 1100.0
Link 215,	X = 13.0	Y = 1100.0
Link 217,	X = 16.5	Y = 1100.0
Link 1,	X = 20.0	Y = 1100.0
Link 56,	X = 26.0	Y = 1100.0
Link 609,	X = 29.5	Y = 1100.0
Link 808,	X = 33.0	Y = 1100.0
Link 115,	X = 39.0	Y = 1100.0
Link 109,	X = 42.5	Y = 1100.0
Link 438,	X = 46.0	Y = 1100.0

(Note that the link numbers are random; this is to illustrate a fragment of a LinkSprint file, not a complete file.)

In this case, each group of three links is repeated with a pitch of 13µm. Within each group, however, the pitch is 3.5µm. Although one could write a LinkSprint file containing 4 groups with 3.5µm pitch, a higher throughput would be achieved if three interleaved groups with a pitch of 13.0 µm were defined instead. An optimized fragment of the LinkSprint file comprising these links would be:

```
0.0    1100.0    13.0  0.0    4
{ 17 215 56 115 }
42.5   1100.0   -13.0  0.0    4
{ 109 609 217 21 }
7.0    1100.0    13.0  0.0    4
{ 313 1 808 438 }
```

These links are illustrated in Figure 3.

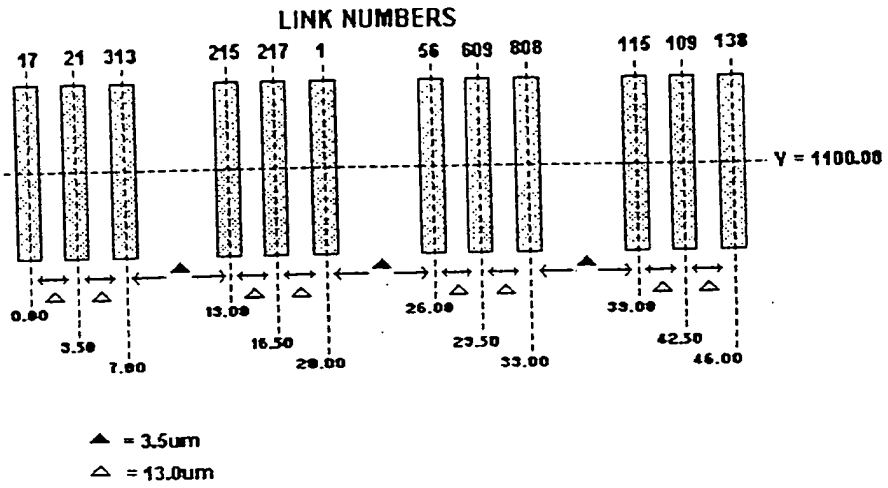

FIGURE 3.

Phantom links are used in the case where links are laid out in a pattern where the distances between them are an integer multiple of a common pitch. For example, if the links are:

```
Link 191,   X = 17.5   Y = -1670.3
Link 192,   X = 17.5   Y = -1674.8
Link 72,    X = 17.5   Y= -1683.8
Link 99,    X = 17.5   Y = -1688.3
Link 103,   X = 17.5   Y = -1701.8
Link 88,.   X = 17.5   Y = -1710.8
```

The links in this case are all on a y-pitch of 4.5µm yet there are 3 "missing" links. The LinkSprint file fragment would look like this:

```
17.5 -1670.3   0.0   -4.5   10
{ 191 192 0 72 99 0 0 103 0 88 }
```

The zeros in the link number list mean simply that "there is no link there". These links are illustrated in Figure 4.

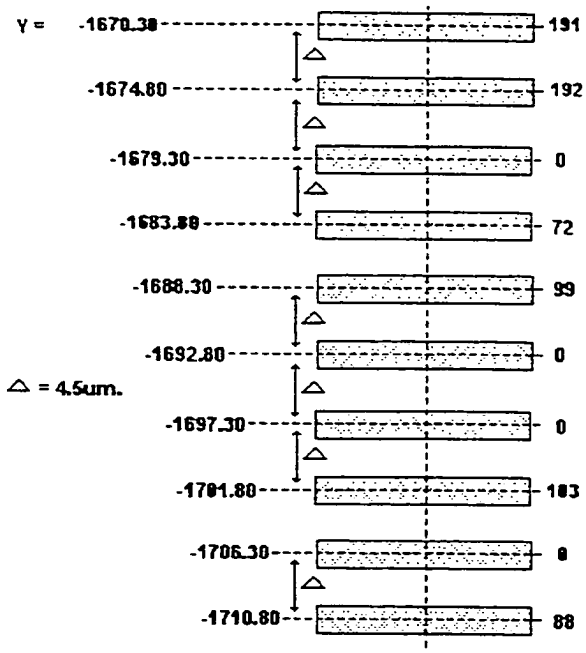

FIGURE 4.

Single links are expressed as a LinkSprint group containing one link. For example, consider one link:

Link 722, X = -1567.8 Y = 191.4

The LinkSprint file fragment would look like this:

```
-1567.8   191.4   0.0   0.0   1
{ 722 }
```

Note that both the x-pitch and y-pitch is 0.0. This is OK as long as only one blast per link will ever be used on this device. If however one wishes to implement multiple blasts per link, PROCESS_TOOL must be made aware of the link's orientation, i.e., whether its long direction is parallel to the X-axis or the Y-axis. PROCESS_TOOL assumes that links with a zero x-pitch and a non-zero y-pitch have their long dimension parallel to the X-axis. Conversely, links with a zero y-pitch and a non-zero x-pitch have their long dimension parallel to the Y-axis. The value of the pitch should be 1.0µm in the axis orthogonal to the link's orientation.

The same link with its long dimension parallel to the Y-axis would be expressed as:

```
-1567.8   191.4   1.0   0.0   1
{ 722 }
```

And similarly, the same link in the other orientation would be:

```
-1567.8   191.4   0.0   1.0   1
{ 722 }
```

TLSI provides a program called "sortlinks" which automatically generates a file in LinkSprint syntax from a customer-supplied list of links. The customer's link file must be in this format:

```
N
X1 Y1
X2 Y2
X3 Y3
X4 Y4
...
...
XN YN
```

The first line contains the total number of links that the device has and the next N lines contain all the coordinate pairs, i.e., X1 Y1 are the coordinates of link #1, X2 Y2 are the coordinates of link #2, and so on until link #N.

If the ASCII file (*input_file*) containing these links looked like this:

```
5
1545.7 338.92
1558.2 338.92
1570.7 338.92
1583.2 338.92
1595.7 338.92
```

the sortlinks program would generate a file as follows:

```
5 1
1545.7 338.92  12.5 0.0  5
{ 1 2 3 4 5 }
```

To run sortlinks one must type:

sortlinks *input_filename output_filename*

at the system prompt. The *output_filename* will be in a format compatible with PROCESS_TOOL. Sortlinks in general will try to find interleaved patterns and patterns with phantom links. It cannot, however, always generate the most optimal grouping. In a high-throughput environment application with many links per die that must be cut, it is recommended that one manually "tweak" the output of sortlinks or generate the file entirely by hand.

For more information on sortlinks, contact TLSI's "M" Applications Engineering group.

In general, to optimize throughput one must:

1) Minimize the total number of LinkSprint groups.
2) If the number of phantom links between real links times the pitch exceeds 200.0µm, use two groups instead of one.
3) The ending link of a LinkSprint group should be physically close to the beginning link of the following LinkSprint group.